# Predict commute time to work from census data

Daniele Mansillo
*Politecnico di Torino*
Student id: s319297
s319297@studenti.polito.it

Simone Rogato
*Politecnico di Torino*
Student id: s289863
s289863@studenti.polito.it

***Abstract*—In this report we present a possible solution to address the regression problem of predicting the commute time to work based on comprehensive census data. The solution compares several regression models achieving promising results and explores the possible underlying implications of socioeconomic disparities in the result.**

## I. Problem overview

The goal of the proposed task is to analyze a dataset describing the travel time to work based on an individual's personal information in order to predict the travel time to work of unlabeled rows and explore underlying socioeconomic disparities to inform evidence-based policy decisions. The provided dataset consists in two different sections:

- A development set of 104,642 instances characterized by personal information concerning occupation, education, sex, family status, sensitive information like ethnicity and origin and the time of departure from home and arrival at work with the target variable, Travel time to work (JWMNP).
- An evaluation set of 26,160 instances, structured as the previous one but without the numerical target variable to predict (JWMNP - Travel time to work).

The target numerical variable is in range 0, 200 and its distribution is concentrated around the multiples of 5, giving us a hint that some sort of approximation could have been used during the data collection, in fact, counting all the rows whose value of JWMNP is not a multiple of 5 we obtain only 10,233 rows, less than 1/10 of the development set.

Most of the features present in the dataset are categorical features, some of them with hundreds of possible different values, this represents a major issue to tackle when building our prediction model.

There are no missing values, there are some rows with the same values but with different Ids and we will suppose they represent a common case.

But, in Fig. 1 we can see how some features have a remarkably high correlation, so, in order to reduce the number categorical features, we decided to exclude some of them from our model to simplify the representation.

## II. Proposed approach

### A. Preprocessing

As the first step we considered the departure time and arrival time since they were the columns that were the most likely to determine the Travel time to work. Since they both represented time ranges between 12.00 AM and 12.00 PM we decided to convert each value to get two columnns from each field, the beginning of the time range and the end of the time range. We chose to represent these new columns as the minutes elapsed from the midnight (integer value from 0 to 1440). After this operation we have 4 new columns:

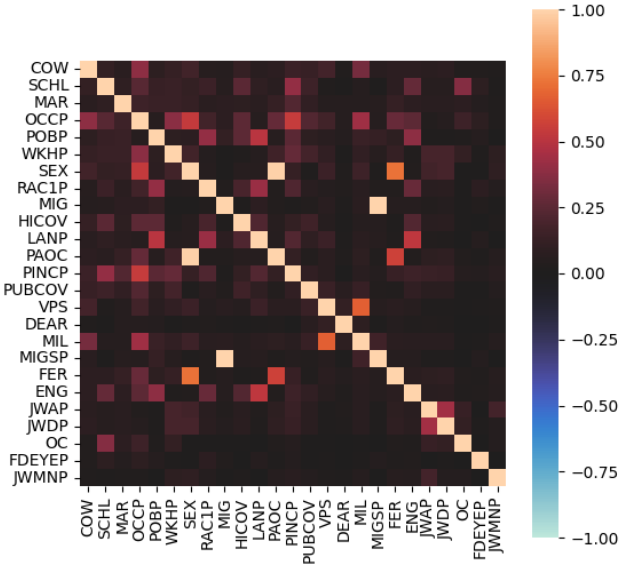


Fig. 1. correlation matrix.

- **JWDP_B**: the starting time in minutes of the departure window
- **JWDP_E**: the end time in minutes of the departure window
- **JWAP_B**: the starting time in minutes of the arrival window
- **JWAP_E**: the end time in minutes of the arrival window.

We drop the original JWDP and JWAP and from these new columns we calculate 5 new ones:

- **JWMNP_B**: the difference between the starting time of the arrival and the departure window (JWAP_B − JWDP_B)

- **JWMNP_E**: the difference between the end times of the arrival and the departure window (JWAP_E−JWDP_E)
- **JWMNP_B_E**: the difference between the end time of the arrival window and the starting time of the departure window (JWAP_E − JWDP_B)
- **JWMNP_E_B**: the difference between the starting time of the arrival window and the end time of the departure window (JWAP_B − JWDP_E)
- **JWMNP_a**: the difference between the end time of the arrival window and the starting time of the departure window divided by two ((JWAP_E − JWDP_B)/2)

The first two columns in the list are very significant for the prediction, out of 104,642 rows in the development set, in 91,220 cases JWMNP = JWMNP_B and in 73321 cases JWMNP = JWMNP_E.

We then reduce the cardinality of the **OCCP** column by mapping the code to the name of the occupation and keeping only the first 3 characters which represent the macro-category of the occupation, in this way we reduce the distinct values from 529 to 25.

To reduce the cardinality of the **POBP** column we group the countries by continent, it gives us a coarse precision but it also reduces the number of distinct values from 219 to 5. We then applied the same process to the **MIGSP** column reducing the number of distinct values from 96 to 6.

We also reduce the number of different values in the **SCHL** column by grouping some fine grained categories together reducing the cardinality from 24 to 10.

Also, analyzing the correlation matrix, available in Fig. 1 we exclude some columns that are fairly explained by other columns:

- **MIG**: it has a correlation value of 1.00 with the column MIGSP which conveys a more complete information;
- **PAOC**: it has a correlation value of 0.99 with the column SEX but provides a more elaborate information which does not seem relevant in the analyzed case;
- **FER**: it has a correlation value of 0.73 with the column SEX and we drop for the same reason above;
- **VPS**: it has a correlation value of 0.67 with the column MIL which is more general;

For the column **LANP**, which contained 119 different values, we decided to keep only the 5 most frequently spoken languages ignoring all the other minority languages; we opted for 5 because between the 5th and the 6th there is a remarkable drop in frequency.

We then apply One-Hot Encoding to all the categorical features and we use a Standard Scaler on all numeric features.

Through all these preprocessing steps we managed to reduce the number of columns from more than 1500 (in case we applied One-Hot Encoding to the raw data) to just 91, excluding the target column we are now left with 90 columns.

### *B. Model selection*

After a preliminary analysis we decided to take into consideration the following regression models: BaggingRegressor, ExtraTreesRegressor, GradientBoostingRegressor and RandomForestRegressor. The main reasons that led us to this choice are the following:

- **Ensemble Methods**: All four of these regressors are ensemble methods, which means they combine the predictions of multiple individual models to make more accurate predictions. This ensemble approach can help mitigate the challenges posed by high dimensionality and sparsity. [1]
- **Reduced Overfitting**: High-dimensional datasets are prone to overfitting. The methods we presented involve aggregating the predictions of multiple base models, which helps reduce overfitting and improve generalization to new data.
- **Feature Selection**: Random Forest [2] and ExtraTreesRegressor [3] perform feature selection implicitly. They select a random subset of features at each split in the decision tree, which can help in identifying important features while ignoring noisy or irrelevant ones, which is particularly useful in high-dimensional datasets.
- **Robustness to Noise**: Ensemble methods are generally robust to noisy data, and in high-dimensional datasets, there is often a lot of noise. By aggregating predictions, these methods tend to smooth out the impact of noisy data points.
- **Handling Sparsity**: These ensemble methods can handle sparse data reasonably well. Decision trees, which are the base models for these regressors, naturally deal with sparse data by selecting features at each split.
- **Parallelization**: Many of these regressors, including Random Forest and ExtraTreesRegressor, can be easily parallelized, which can significantly speed up training on high-dimensional datasets.

We decided to leave out SVM and Logistic and Linear Regression since their performance suffer immensely from the curse of dimensionality providing poor generalization and long training time.

After training a model for each one of the proposed regressors our choice fell on the **GradientBoostingRegressor** [4] since it showed the best performance and also a significantly shorter training time.

### *C. Hyperparameters tuning*

Using the GradientBoostingRegressor with the default parameters we obtained an $r^2$ score of 0.9967, which is already a remarkable result, but, in order to get the best out of this model, we decided to apply the following GridSearch to our selected algorithm:

| Parameter | Values |
|---|---|
| learning_rate | 0.1, 0.01, 0.2 |
| n_estimators | 50, 75, 100, 200 |
| criterion | squared_error, friedman_mse |
| max_depth | 3, 4, 5, 6 |

## III. RESULTS

After the GridSearch the best configuration for the model turned out to be the following one:

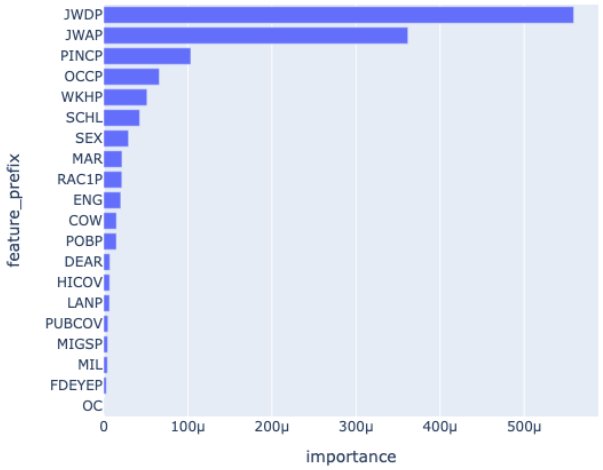


Fig. 2. feature importances sum per category.

| Parameter | Value |
|---|---|
| learning_rate | 0.2 |
| n_estimators | 75 |
| criterion | friedman_mse |
| max_depth | 6 |

Through the hyperparameter tuning we managed to obtain an increase in the performance, from 0.9967 to 0.9971.

## IV. Discussion

Let us now analyze the feature importance to investigate which columns contributed the most to the result and to discover if there is some kind of evidence revealing any underlying social disparity that influenced the result.

We observe how some features are almost totally irrelevant, like OC, FDEYEP, MIL, MIGSP, PUBCOV.

As expected the most relevant features are the ones we calculated in the preprocessing step, JWMNP_B, JWMNP_E. JWMNP_B_E, JWMNP_E_B, JWMNP_A but, also some features play a very minor role like PINCP, OCCP, WKHP, SCHL, SEX which, except for SEX, are features regarding work and economical disparities, more than social disparities. Only SEX, MAR and RAC1P among the features regarding disabilities, race, origin and sex seem to have some kind of relevance while the other ones are almost negligible

In Fig. 2 we can see a visualization of the sum of feature importances from which we removed all the generated JWMNP columns which accounted for more than 99.8% of the feature importances.

## References

[1] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] P. Geurts, D. Ernst, and L. Wehenkel, "Random decision forests," in *Proceedings of the 2006 European conference on machine learning and knowledge discovery in databases*, pp. 396–407, 2006.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.